4K－9　ビジュアルモデリングのためのメタモデル作成支援方法の検討とその実現

高橋 大輔　安間 その美　上田 賀一

茨城大学工学部情報工学科

1　はじめに

対象とする問題領域のモデルを作成することは問題の理解や解決に対して非常に有効な手法であると思われる。

本研究ではモデル記述にメタモデルの概念[1]を導入し、ビジュアルモデリング時の表示部分と内部情報との関係について検討した上で、メタモデルを表現するための枠組としての概念を定義した。

さらにメタモデルを簡潔に記述するためにメタモデル記述言語で記述し、メタモデル設計者の負担を軽減するためのメタモデル作成支援ツール(メタモデラ)を実現した。

2　モデル

モデルとは理論の探求や問題の解決のため、対象とする問題の環境、枠組や本質を抽象化したものである。モデルを作成することで問題解決に必要な側面のみを捉えることが可能となり、複雑さを減少させ問題領域の全体構造の把握が容易になる。

モデルの記述法としてはモデル記述用の言語による記述や図を利用して記述する方法が存在する。ソフトウェア工学分野では状態遷移図やデータフロー図などのように図式表記されることが多い。これは言語表記よりも直観的、論理的に全体構造を把握しやすく、情報の伝達手段としても有用であるためであると思われる。

3　メタモデル

モデルはモデルの構成要素(コンポーネント)を配置し、組み合わせることにより作成される。モデルの構成要素はドメイン毎に1つのまとまりとして管理・利用されるものとする。このドメイン毎の構成要素の集合とそれらに対する制約をメタモデルとし、メタモデルに含まれる構成要素(メタコンポーネント)を利用してモデルを記述する。

以下ではメタモデルを定義するためにモデルの表現方法をその表示的側面と意味的側面から検討する。

表示的側面

状態遷移図やデータフロー図など図式表記されるモデルは形状や線種に多少の差異はあるが、Node(節点)とArc(弧)で構成されているものがほとんどである。そこでNodeとArcで構成されているモデルをNAモデルと呼び、モデルの表示にNAモデルを利用することとする。

Nodeは円や四角、または絵などで表示され、Arcは実線や破線など直線、または曲線によって表示される。

意味的側面

メタモデルの意味表現にはERモデルを利用する。ERモデルではEntity(実体)とEntity間のRelationship(関連)という2つの観点を用いて実世界の意味構造や制約条件をより直観的かつ自然に表現することが可能である。

ERモデルでは動的側面を表現する能力はなく検証可能なモデル記述が行なえない。そこでERモデルにオブジェクト指向の概念を採り入れ、動的側面を含めるようなOERモデルとした。

OERモデルではEntityとRelationshipはオブジェクトとして扱われる。

表示的側面と意味的側面との関係

表示と意味で異なるモデルを利用するのでモデル間の関係を定義する。Entityを表示するにはNodeを利用する(Entity-Node)。Relationshipを表示するにはArcを利用するが、Arcだけでは表示できないRelationshipも存在する。このためRelationshipの表示にはArcを利用するもの(Relationship-Arc)とNodeとArcを利用するもの(Relationship-Node-Arc)を用意する。

Study of meta-modeling concept for visual modeling
Daisuke Takahashi, Sonomi Anma, Yoshikazu Ueda
Department of Computer and Information Sciences, Faculty of Engineering, Ibaraki University

4 メタメタモデル

メタモデルはモデルを記述するための要素の集合であるとした。上で述べたERやNAの概念はメタモデルを記述するための概念であるのでメタメタモデルにそれらの概念を持たせることにする。

その他にメタメタモデルにはメタモデル記述時に必要な以下の機能を持たせている。

メタコンポーネントの継承

既存のメタコンポーネントから継承を使って新たなメタコンポーネントを作成できる。

事前・事後条件

Entityはメッセージを受け取るとアクションを起動する。このアクションを起動する前と起動した後の制約条件を設定できる。

多重度の記述

あるコンポーネントが同一のメタコンポーネントから生成されたコンポーネントと関連する数を多重度[2]といい、その値はメタモデル記述時に設定でき、モデル構築時に検証できる。

メッセージの型

Entity間のメッセージ送信に3種類の型[3]を持たせた。以下にそれぞれのメッセージの型を示す。

過去型 メッセージに対する返答はなく、送信後すぐに次の処理へ進む

現在型 メッセージに対する返答があるまで次の処理をしない

未来型 メッセージに対する返答があるが、返答を待たずに次の処理へ進む

5 メタモデル記述言語

メタメタモデルで規定した項目に基づいてメタモデルを記述するための言語を作成した。この記述言語によりEntity・Relationshipおよびそれらの属性を記述することができる。言語はC++を拡張したもので、記述されたメタモデルはトランスレータを通してモデラやシミュレータなど他のツールで利用できる形式に変換される。

```
ENTITY sink {              // Entity の宣言
  ICON "sink.xbm";         // Node として表示する絵
  NAME "sink";             // 管理用の名前
  int  count;              // ユーザ定義の変数
  ACTION recieve;          // アクションの宣言
};

ACTION sink::recieve {
  ...                      // アクションの処理を記述
}

RELATIONSHIP flow {        // Relationship の宣言
  WIDTH 1;                 // Arc の線の幅
  LINESTYLE REAL;          // 線種の指定
  ARROWTYPE SINGLE;        // 矢印の型
  CARDINAL source:sink = 1:1<  // 多重度
};
```

図1:メタモデル記述言語による記述例

6 メタモデラ

メタモデル設計者の負担を軽減するためにメタモデラを作成した。メタモデラはUnix上でTcl/Tkで実装されており、メタモデルはディレクトリ単位で、メタオブジェクトはファイル単位で扱われる。

メタモデラは以下の機能を持つ。

- Entity間のRelationshipの視覚的な編集
- メタコンポーネントの属性およびアクションの編集
- アクションの実行前後の制約の編集
- 編集されたメタモデルのメタモデル記述言語による出力
- クラス階層のブラウザ

7 おわりに

以上、モデルを構築するためのメタモデルについて検討した。今後、メタモデルと、メタモデルに基づきモデル構築するためのモデラやそのモデルを駆動させるシミュレータとの関連について検討していきたいと考える。

参考文献

[1] 清水俊吾：“プロトタイピングのためのモデル記述言語と支援環境”, 情報処理学会, ソフトウェア工学研究会, Vol.93, No.84, pp.49-56,1993

[2] James Rumbaugh 他 (羽生田監訳)：“オブジェクト指向方法論 OMT”, トッパン

[3] 所真理雄, 松岡聡, 垂水浩幸：“オブジェクト指向コンピューティング”, 岩波書店